Monitorització del consum elèctric amb la Raspberry Pi

Grau en Enginyeria Electrònica Industrial i Automàtica

Autor: Josep Borràs Milla

Tutor: Carles Mateu Piñol

Universitat de Lleida
Escola Politècnica Superior

Juny del 2017

AGRAÏMENTS

En primer lloc, m'agradaria agrair per tot l'esforç i preocupació al meu pare i la meva mare, pel suport incondicional que m'han aportat en tot moment durant tot aquest projecte.

En segon lloc, i tant o més important que els meus pares, el meu tutor, el Carles Mateu, una persona que a part de resoldre'm qualsevol dubte dels molts que he anat trobant, m'ha guiat també en algun moment quan em trobava perdut, mostrant-me sempre el seu interès en la realització d'aquest projecte.

Finalment, als companys i amics del grau de Enginyeria Electrònica Industrial i Automàtica, dels quals he tingut el plaer d'aprendre molts coneixements, i també m'han anat orientant durant tot aquest camí.

Índex General

Índex de figures

Índex de taules

1. Introducció

1.1 Descripció del projecte

El present projecte tracta de la construcció d'un prototip d'analitzador de xarxes elèctriques per a la visualització en temps real dels consums elèctrics en diferents punts. A més a més cap l'opció de creació d'unes gràfiques amb les dades recollides a temps real.

1.2 Motivació

La motivació d'aquest projecte apareix quan estava finalitzant les meves practiques curriculars en una empresa del sector energètic.

L'empresa em va demanar que busques diferents aparells de varis proveïdors per analitzar xarxes elèctriques per a muntatges en diferents llocs, com naus industrials, poliesportius, ...

Els projectes es van realitzar però al meu semblar, vaig veure uns preus desorbitats que podia aconseguir una forma molt més econòmica de realitzar aquestes tasques. Una d'elles era utilitzant la Raspberry Pi, eina que havia treballat el darrer curs i m'havia sorprès del seu funcionament i la seva amplia varietat d'aplicacions.

1.3 Objectius

Els objectius d'aquest projecte són:

- Realitzar un prototip **més econòmic i competitiu** que altres del mercat.
- Oferir **senzillesa i facilitat** per a l' implementació de qualsevol persona.
- Fàcil **visualització de dades i la seva recopilació** per a llargs períodes de temps.

2. Característiques tècniques

2.1 Raspberry Pi

2.1.1 Introducció

La Raspberry Pi es tracta d'una diminuta placa base de 85x54 mm (una mica més gran que un paquet de tàbac) en el qual es troba un chip Broadcom BCM2835 amb processador ARM fins a 1 GHZ de velocitat, GPU VideoCore IV i fins a 512 Mbytes de memòria RAM. El seu preu deu estar sobre els 40 euros una de les raons per la qual s'explica la seva popularitat.

Per a que funcioni sol fa falta que nosaltres mateixos afegim un mitja d'emmagatzematge (per exemple una targeta de memòria microSD), endollar-lo a la corrent amb qualsevol tipus de carregador microUSB.

La fundació Raspberry Pi posa a disposició a la seva pàgina web Raspbian, una distribució Linux basada en Debian, però també es pot recórrer a altres distribucions que la comunitat ha desenvolupat per altre finalitats.

En funció del model escollit de la Raspberry, disposarem de diferents característiques en el processador, memòria RAM, ... i més o menys opcions de connexions, però sempre hi haurà almenys un port de sortida HDMI, un de RCA, un minijack de àudio, i un port USB 2.0.

En referència a la connexió de xarxa podem disposar d'Ethernet per connectar un cable RJ-45 directament del router, o recórrer a un adaptador WIFI que es connecta al port USB, o bé, en el millors dels casos i si es disposa del últim model fins al moment, la Raspberry Pi 3 que ja ho porta incorporat.

Figura 1: Raspberry Pi

Per a l'emmagatzematge, Raspberry Pi recomana utilitzar una targeta SD amb una capacitat mínima de 4GB i de clase 4.

Per connectar la nostra Raspberry Pi a un monitor o televisió, necessitarem un cable HDMI o, si no disposem de tal entrada de vídeo, un cable HDMI a DVI. També es pot recórrer en el seu lloc a la sortida analògica RCA.

Pel que fa al teclat i ratolí el més senzill és adquirir un conjunt sense fils que connectarem mitjançant un únic adaptador USB.

2.1.2 Primeres passes a la Raspberry Pi

Després de la instal·lació de Raspbian, el primer que hem de fer es configurar connexió a Internet per poder actualitzar.

Amb la comanda **raspi-config** podrem configurar diversos paràmetres de la nostra Raspberry com per exemple, l' idioma, l' idioma del teclat i el tipus, l'hora i la zona horària, ...

Aleshores s'ha d'actualitzar la versió de qualsevol software instal·lat. Des d'un terminal executem la següent comanda:

```
sudo apt-get update
```

Després d'això tindrem una llista de les versions disponibles de tot el software instal·lat, no sol del sistema operatiu. Aquesta llista no es veu, sol que al acabar el procés anterior sabem que s'ha actualitzat. Ara actualitzem realment:

```
sudo apt-get upgrade
```

Un cop actualitzat el software, es recomanable reiniciar la Raspberry.

2.1.3 Consola i comandes bàsiques per la seu ús

A continuació veurem algunes comandes bàsiques per utilitzar la Raspberry en mode de text a través de la consola.

- **Execució de comandes com a super usuari**
  sudo

- **Menú de configuració**
  raspi-config

- **Apagar el sistema**
  sudo poweroff
  sudo shutdown –h now

- **Reiniciar el sistema**
  sudo reboot
  sudo shutdown –r

- **Actualitzar el sistema**
  sudo apt-get update
  sudo apt-get upgrade

- **Crear directori o carpeta**
  mkdir *carpeta*

- **Esborrar directori o carpeta**
  rmdir *carpeta*

- **Esborrar directori o carpeta no buits**
  rm –rf *carpeta*

- **Desplaçar-se a un directori**
  cd *carpeta*

- **Saber en quin directori estem**
  pwd

- **Mostrar el contingut d'una carpeta**
  ls

- **Mostrar el contingut d'una carpeta amb informació**
  ls –l

- **Anar al directori anterior**
  cd ..

- **Anar al directori *home***
  cd

- **Anar al directori arrel**
  cd /

- **Crear o editar un fitxer**
  nano *fitxer*

- **Esborrar un fitxer**
  rm *fitxer*

- **Copiar un fitxer a una carpeta**
  cp *fitxer/carpeta*

- **Moure un fitxer a una carpeta**
  mv *fitxer/carpeta*

2.1.4 Terminals GPIO

GPIO (General Purpose Input/Output) és, com el seu propi nom indica, un sistema d'E/S (Entrada/Sortida) de propòsit general, és a dir, una sèrie de connexions que es poden usar com a entrades o sortides per a usos múltiples. Aquests pins estan inclosos en tots els models de Raspberry Pi.

Els GPIO representen la interfície entre la Raspberry Pi i el món exterior .

Tots els pins són de tipus "unbuffered", és a dir, no disposen de memòries intermèdies de protecció, així que s'ha de tenir cura amb les magnituds (voltatges, intensitat, ...) quan es connectens components a ells per no danyar la placa. No tots els pins tenen la mateix funció

- **Pins d'alimentació:** es pot apreciar pins de 5v, 3v3 (limitats a 50mA) i terra (GND o Ground), que aporten alimentació a aquests voltatges per als circuits. Poden servir com una font d'alimentació, tot i que també es poden utilitzar altres fonts (piles, fonts d'alimentació externes, etc.).

- **DNC (Do Not Connect):** són pins que de moment no tenen funció, però en futures implementacions són utilitzats per a altres fins. Per això només es trobarem a models més primitius de la Raspberry Pi. En les actuals plaques han estat marcats com a GND.

- **GPIO normals:** són connexions configurables que es poden programar en projectes.

- **GPIO especials:** dins d'aquests es troben alguns pins destinats a una interfície UART, amb connexions TXD i RXD que serveixen per a comunicacions en sèrie, com per exemple, connectar amb una placa Arduino. També podem veure altres com SDA, SCL, MOSI, MISO, SCLK, CE0, CE1, etc ...

Figura 2: Terminals GPIO

2.2 Sensor de corrent SCT-013

Els sensors de la sèrie SCT-013 són sensors que treballen com a transformadors, el corrent que circula pel cable que desitgem mesurar actua com el debanat primari (1 espira) i internament té un debanat secundari que depenent del model poden tenir fins a més de 2000 espires.

La quantitat d'espires representa la relació entre corrent que circula pel cable i la que el sensor ens lliura, aquesta relació o proporció és la que diferencia entre els diferents models de sensors SCT-013, addicionalment poden tenir una resistència de càrrega a la sortida d'aquesta forma en lloc de corrent es treballa amb una sortida voltatge

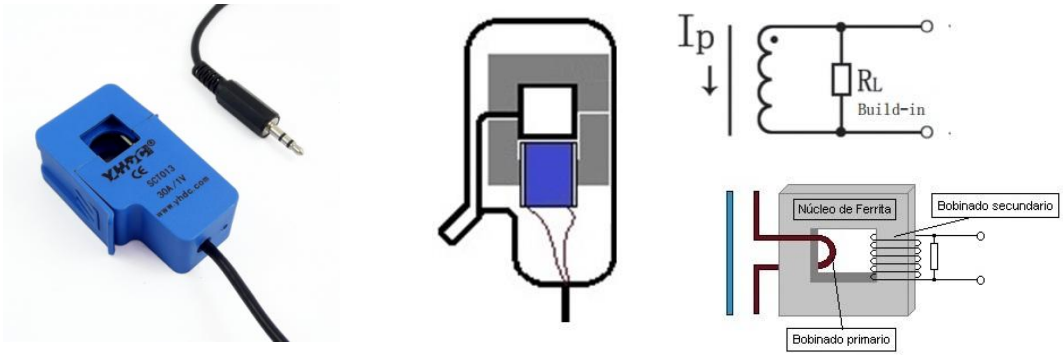

Figura 3: Sensor de Corrent SCT013 i esquema

A aquest tipus de sensors se'ls coneix com Sensors CT (Current transformers. Pel nucli només ha de travessar una sola línia, si passem per exemple els dos cables d'una connexió monofàsica, la nostra lectura serà 0 , ja que els cables tenen corrents oposats.

Un avantatge de SCT-013 és que no necessitem interrompre (tallar o desempalmar) el cable que anem a mesurar, això passa perquè igual que una pinça amperimètrica té el nucli partit.

3. Monitorització del consum elèctric

En aquest apartat veurem alguns tipus d'aparells del mercat semblants al que nosaltres volem desenvolupar, així al final del projecte podrem comparar-los.

3.1 Pack Efergy P2 + Engage Hub (Trifàsic)

El Efergy Engage Hub és un sistema que es connecta directament al router amb el cable Lan. Una vegada donats d'alta a la web es podrà saber el consum elèctric en temps real.

Incorpora un monitor portàtil per veure el consum a temps real en la seva pantalla on hi ha tres modes de visualització: cost, CO2 i kwh. És pot alimentar amb piles o per un adaptador elèctric.

El seu funcionament es ben senzill, consta de 3 pinces que es poden controlar una instal·lació trifàsic o bé tres de monofàsiques. La informació del consum real s'envia de forma inal·làmbrica al engage Hub, que al mateix temps aquest està connectat a Internet via el cable de ethernet.

Figura 4: Pack Efergy P2

3.2 Mirubox v2

Mirubox es un mesurador de consum elèctric que permet una mesura elèctriques amb una corrent màxima de 100A.

A través de connexió Wifi permet enviar i registrar les dades al nuvol, pel seu posterior analisis i gestió. Utilitza una plataforma WEB gratuïta per visualitzar les dades ja sigui per PC o aplicació mòvil.

Figura 5: Mirubox v2

3.3 EmonPi

El EmonPi es un sistema de monitorització de l'energia basat en una senzilla instal·lació amb enviament de les dades a la xarxa per Wifi o Ethernet. També pot mesurar la temperatura i la interfície amb mesuradors adequats per això.

Esta basat en un sistema de Raspberry Pi tot introduït dins d'una carcassa adaptat per a les seves conexions. Disposa d'una pantalla LCD molt simple de dos linies, per a la funcionalitat de la configuració a la xarxa i per observar que tot funciona correctament

Figura 6: EmonPi

4. Disseny

4.1 Requisits del projecte

Una de les primeres coses abans de realitzar el projecte és tenir una visió clara del que volem fer, les necessitats, desitjos i expectatives contingudes dins d'ell

4.1.1 Requisits funcionals

Son els requisits ens els quals l'usuari pot intervenir

- Lectura del consum elèctric: L'usuari podrà connectar un, dos, o fins a tres sensors simultanis al quadre elèctric per mesurar la potència de cada línia.
- Elecció tipus de lectura: L'usuari podrà escollir entre una lectura monofàsica o trifàsica.
- Elecció dels paràmetres de lectura i emmagatzematge: L'usuari podrà definir el temps transcorregut entre mesures, el nom de la base de dades on es guardaran les lectures, i el nom que vulgui assignar a cada sensor.
- Configurar wifi: Si l'usuari ho desitja pot configurar una connexió wifi per poder comunicar les dades amb una plataforma web.
- Configuració de la plataforma web: És podrà configurar una plataforma web a mida on l'usuari podrà veure les dades representades gràficament, en taules, obtenir els màxims, mínims, ...

4.1.2 Requisits no funcionals

- Suport: El sistema tindrà un suport tècnic per qualsevol dubte o problema del usuaris.
- Usabilitat: Tant el dispositiu com la plataforma web seran senzills i intuïtius.
- Rendiment: El rendiment del sistema serà el més òptim possible
- Baix cost: El producte tindrà el menor cost possible.

4.2 Disseny del sistema

El sistema funcionarà de la següent forma: amb el sensor connectat al quadre la Raspberry rebrà les dades, les mostrarà en una pantalla (solament les instantànies), les emmagatzemarà i mitjanant connexió a xarxa podrem accedir a les dades i visualitzar-les en un ordenador, en forma de gràfic, taula, ...

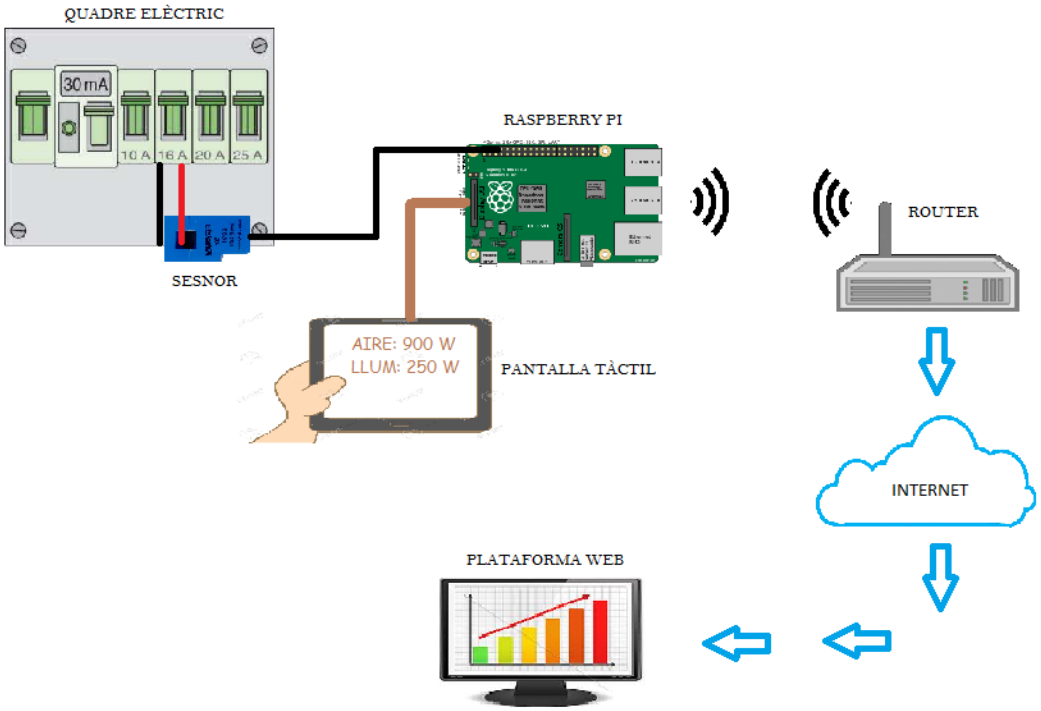

Figura 2: Disseny gràfic del sistema

4.3 Comunicació física

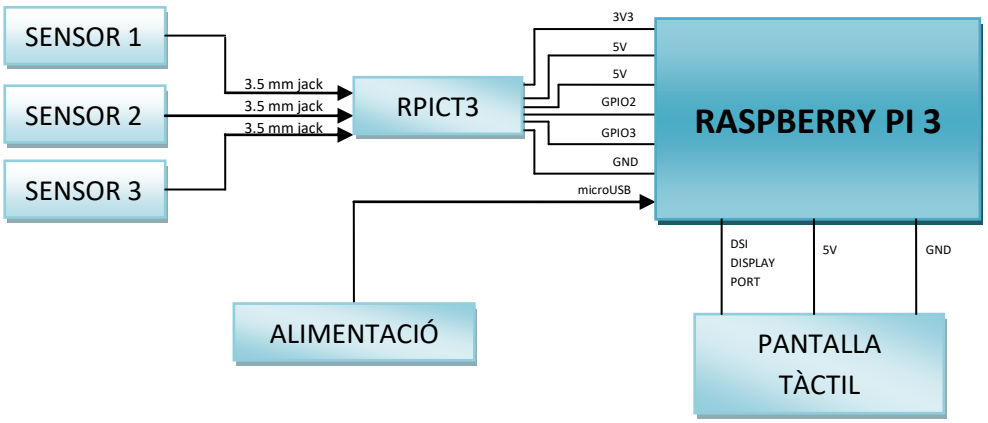

Figura 8: Esquema de les connexions del sistema

4.4 Disseny electrònic

Per la part electrònica del projecte utilitzarem diversos elements importants i altres opcionals que anomenarem a continuació.

Els elements fonamentals són: la **Raspberry Pi 3** (amb la 2 model B també seria suficient) que llegirà els sensors, guardarà les dades, i les mostrarà per pantalla, el **transformador de corrent** per alimentar a la Raspberry Pi, els **sensors de corrent** que mesuraran la corrent, el **RPICT3** que és un mòdul entre els sensors i la Raspberry, i el **mòdul wifi** per la transferència de dades (en el cas de la Raspberry Pi 3 no fa falta ja el porta incorporat).

4.4.1 Raspberry Pi

S'ha escollit la Raspberry Pi 3 enfront a la possibilitat d'utilitzar un arduino ja que s'ha valora't l'opció de mostrar les dades per una pantalla, l'emmagatzematge de la base de dades en el propi microcontrolador, i l'adaptador RPICT que explicarem més endavant, en un arduino hagués estat pràcticament impossible combinar tantes opcions. Enfront a l'elecció del model, ens hem decantat per la Raspberry Pi B 3 ja que te un preu molt similar a la 2 i a canvi te molta més potència i incorpora el mòdul wifi per tant no cal utilitzar un mòdul amb USB.

Taula 1: Característiques Raspberry Pi 3

RASPBERRY PI 3	CARACTERÍSTIQUES
SoC	BC2837
CPU	Quad Cortex A53 @ 1.2GHZ
CONJUNT D'INSTRUCCIONS	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
EMMAGATZEMATGE	Micro-SD
ETHERNET	10/100
WIRELESS	802.11n / Bluetooth 4.0
VIDEO OUTPUT	HDMI / Composite
AUDIO OUTPUT	HDMI / Headphone
GPIO	40
PREU	37€

4.4.2 Sensors de Corrent

Per mesurar la corrent que passa pels conductors utilitzarem el transformador de corrent de nucli ferromagnètic (pinça) que permet obrir-lo per col·loca'l al voltant del fil de mesura sense necessitat de talla'l.

La decisió d'utilitzar aquest sensor és la explicada anteriorment, no es necessita fer cap tall en conductes per poder mesurar el corrent, i a més la fàcil adaptació al adaptador RPICT que explicarem a més endavant.

La precisió del sensor pot ser de 1-2%, per això és molt important que la pinça estigui ben tancada. Fins i tot un petit forat d'aire pot introduir desviacions del 10%.

Existeixen diversos sensors de la família SCT-013, que poden variar el rang de medició i la forma de sortida. El preu de tots ells esta entre els 4-4,50€,

El model que utilitzarem serà el SCT-013-000 en el qual pot mesurar una corrent màxima de 100A (corrent suficient per la instal·lació d'una vivenda) i sortida d'intensitat de 50mA (100A:50mA).

Figura 9: Sensor de corrent SCT013

4.4.3 RPICT

El RPICT és un adaptador per a la Raspberry que permet connectar varis sensors de corrent, voltatge o temperatura al mateix temps.

Hi ha deferents models del segons la quantitat i el tipus de sensors a conectar:

- PICT2T1 2 entrades de corrent / 1 entrada de temperatura
- RPICT3 3 entrades de corrent
- RPICT3V1 3 entrades de corrent / 1 entrada de voltatges
- RPICT3T1 3 entrades de corrent / 1 entrada de temperatura
- RPICT4V3 4 entrades de corrent / 1 entrada de voltatge
- RPICT4T4 4 entrades de corrent / 4 entrades de temperatura
- RPICT7V1 7 entrades de corrent / 1 entrada de voltatge
- RPICT8 8 entrades de corrent

El que utilitzarem serà el **RPICT3** que ens permet connectar 3 sensors de corrent, el seu preu està al voltant del 14€.

El RPICT7V1 seria una altra opció pel nostre projecte que té un preu aproximat de 40€.

El RPICT3 te diverses característiques, entre elles:

- Interval de lectura de 5 segons (nosaltres més endavant podrem augmentar-lo si es desitja però no disminuir-lo)
- Voltatge de 240 volts (per a càlcul de potència)

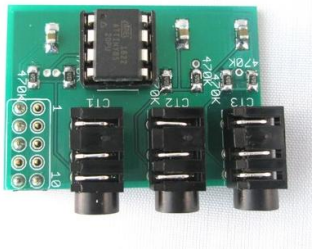

Figura 10: Adaptador RPICT3

Figura 11: Adaptador RPICT7V1

4.5 Pantalla tàctil

Per la pantalla s'ha escollit una pantalla tàctil, ja que amb una normal, el usuari no podria interactuar si no s'introduïa un teclat i ratoli físics que feien més voluminós el sistema, cosa que hem descartat.

El model utilitzat és el Raspberry Pi 7" TouchScreen display, és el més genèric del mercat i aporta una mida suficient gran per a la visualització del usuari.

Com bé diu el nom, te 7" de pantalla amb una resolució de 800x480 i es connecta a la Raspberry Pi mitjançant una placa d'adaptació que s'encarrega de la conversió de potència i senyals. A més és compatible amb un teclat virtual a la pantalla, com ens interessa.

Figura 12: Pantalla táctil 7"

4.6 Base de dades

Les lectures fetes pel sensor seran emmagatzemades cada "x" temps en una base de dades.

En el nostre cas utilitzarem la base de dades **InfluxDB.** El motiu d'utilitzar influxDB es perquè està optimitzat per un emmagatzematge d'alta disponibilitat i recuperació ràpida de dades temporals, es ideal per a la lectura de dades de sensors i el seu anàlisis en temps real.

InfluxDB no té dependéncies externes i proporciona un llenguatge SQL centrada per a la consulta de dades composta de mesures, series i punts. Els valors poden ser nombres enters de fins a 64 bits.

Els punts son indexats pel seu temps i el seu conjunt d'etiquetes.

Les polítiques de retenció es defineixen en un mesurament i controlen com es mostregen i s'esborren les dades.

4.7 Plataforma Web

Per la plataforma web, utilitzarem un software lliure com és el **Grafana** ja que s'adapta a la perfecció amb InfluxDB.

Grafana s'executa com una aplicació web, i des d' allí hi ha diverses opcions per crear gràfics, taules, buscar màxims, mínims... També es poden visualitzar les dades en diferents intervals de temps.

Figura 13: Exemple Grafana

5. Implementació

En aquest apartat del projecte explicarem com s'ha construït el hardware, el firmware, l'aplicació gràfica i l'aplicació web del prototip per al seu correcte funcionament.

5.1 Maquinaria

Seguint les instruccions de connexió de l'apartat 4.3 realitzarem la construcció física del nostre prototip.

Figura 14: Components pel muntatge

Primer de tot muntarem la nostra Raspberry Pi en els suports que ja porta integrats la pantalla tàctil i els seus respectius cargols. Juntament connectarem el cable de vies (integrats a la pantalla tàctil), al port DSI de la Raspberry Pi.

Figura 15: Connexió Raspberry i pantalla tàctil

Seguidament connectarem l'adaptador RPICT3 als GPIOS de la Raspberry Pi, estan ordenats de manera que és poden connectar directe, sense la necessitat de cables i/o ponts de connexió.

Figura 16: Connexió RPICT3 als GPIOS

A continuació sol fa falta connectar els sensors al RPICT3, tants com en tinguem o siguin necessaris.

Figura 17: Connexió dels sensors al RPICT3

I per finalitzar muntem la carcassa de protecció i suport de la pantalla tàctil i de la Raspberry Pi.

Cal dir que s'ha tingut de retallar una part de la carcassa pel lloc on surten els sensors, ja que sinó no era viable la incorporació dels sensors.

Figura 18: Muntatge final

5.2 Firmware

Per a la programació del firmware hem utilitzat el llenguatge Python ja que és el que millor s'adapta a la Raspberry i als sensors.

Hem distribuït el codi en tres arxius per fer-ho més entenedor. Un s'encarrega de les lectures del sensor *(lectures_sensor.py)*, l'altre de la interacció amb la base de dades *(database.py)*, i l'últim el de l'aplicació gràfica *(aplicació_tkinter.py)*.

El programa principal serà el de de l'aplicació gràfica *(aplicació_tkinter.py)*, ja que en funció del que vagi elegint l'usuari s'interactuarà d'una forma o bé d'una altra.

Per tant primer veurem els altres dos programes.

5.2.1 Llibreries utilitzades

El primer que es necessita es carregar totes les llibreries que utilitzarem i els paquets necessaris per a ferho

```
from Tkinter import *
import Tkinter as tk
from tkFont import Font
import ttk
import time
import serial
from influxdb import InfluxDBClient
from influxdb.client import InfluxDBClientError
import datetime
```

Algunes de les llibreries ja estan instal·lades a la Raspberry per defecte, i altres s'han d'instal·lar manualment des de la consola de comandes.

Llibreria per la base de dades:

```
sudo apt-get update
sudo apt-get upgrade
wget
http://ftp.us.debian.org/debian/pool/main/i/influxdb/influxdb_1.0.2+dfsg1-
1_armhf.deb
sudo dpkg -i influxdb_1.0.2+dfsg1-1_armhf.deb
```

Llibreria per a la lectura dels sensors:

```
sudo apt-get install python-serial
```

5.2.2 lectures_sensor.py

Aquest programa solament te un mètode important. El mètode es el *read_sensor* que s'encarrega de llegir la corrent dels sensors i les transforma automàticament en potència gràcies al adaptador RPICT3 (predeterminat a 230V).

L'obtenció dels valors és fa pel port serial on la casa "lechacal" ja dona la informació i una part del codi per fer la lectura de dades. Les dades son representades amb "string" cosa que hem de convertir a "int" pel seu posterior emmagatzematge a la base de dades.

```
def read_sensor(self):
  while True:
    try:
      #Obtenim resposta del port serial
      response = ser.readline()
      #separem resposta en comes
      self.z = response.split(",")
      #eliminem els salts de linia que puguin apareixer
      self.z = [x.replace("\r\n","") for x in self.z]
      #convertim els valors string en float
      self.z = map(float, self.z)
          #Convertim a int ja que influx només accepta int
      for i in range(len(self.z)):
              self.z[i]=int(self.z[i])
      break
    except:
      ser.close()
```

5.2.3 database.py

En aquest programa trobarem tres mètodes, un per crear la base de dades, un altre per veure les bases de dades ja creades, i l'altre per introduir els valors del sensor.

Primer es definiran uns valors que han de tenir les bases de dades de InfluxDB. Podria ser diferent per cada una de elles, però nosaltres establirem el mateix per a totes, així l'usuari no caldrà que interactuí ens aquets valors de poca importància.

```
self.USER = 'root'
self.PASSWORD = 'root'
self.host ='localhost'
self.port = 8086
self.client = InfluxDBClient(self.host, self.port, self.USER, self.PASSWORD)
```

El mètode de crear la base de dades l'únic que farà és obtenir el valor introduït per l'usuari i crear-la.

```
def create_database(self, dbname):

    self.DBNAME = dbname
    self.client.create_database(self.DBNAME)
```

Per obtenir les bases de dades guardades anteriorment InfluxDB té una funció anomenada *get_list_database()* tot i que haurem de fer modificacions als valors obtinguts per a que ens quedi una llista amb només les bases de dades que ho inclourem tot al mètode *get_database()*.

```
def get_database(self):

    self.list_databases = []
    dbs = self.client.get_list_database()

    for x in dbs:
            for key, value in x.iteritems():
             temp = [value]

            data = u", ".join(temp)

            self.list_databases.append(data)
```

Finalment tenim el mètode per introduir les dades, que executarà el mètode *read_sensor()* de l'arxiu lectures_sensor.py, i guardara els valors a la base de dades que hagi elegit l'usuari.

```
def insert_data(self):


    #Introduim les lectures a la base de dades
    metric = "mesures"

    series=[]
    now = datetime.datetime.today()
    pointValues = {
          "measurement": metric,
          "fields":{
              "Pinça 1": self.powerserver.z[0],
              "Pinça 2": self.powerserver.z[1],
              "Pinça 3": self.powerserver.z[2]
           },
      }

    series.append(pointValues)

    self.client = InfluxDBClient(self.host, self.port, self.USER, self.PASSWORD,
                                 self.DBNAME)
    retention_policy = 'awesome_policy'
    self.client.create_retention_policy(retention_policy, '20w', 3,
                                        default=True)
    self.client.write_points(series, retention_policy=retention_policy)
```

5.2.3 aplicació_tkinter.py

Per realitzar l'aplicació gràfica utilitzarem la llibreria Tkinter, que s'utilitza per crear finestres on hi podem afegir text, imatges, botons, etiquetes etc.

El format del programa està distribuït en mètodes principals on cada mètode representarà una finestra de l'aplicació. Un mètode serà cridat per un altre i així es canviarà de finestra.

Es pot veure el codi del programa en l'annex, i en el següent punt la distribució de les finestres de l'aplicació.

5.3 Aplicació gràfica

L'aplicació gràfica és on l'usuari podrà interactuar i variar alguns paràmetres per la lectura dels sensors: quantitat de sensors, base de dades utilitzada, temps entre lectures, ...

Està feta d'una forma molt simple per a que no pugui crear cap tipus de confusió i tot tipus d'usuari la pugui utilitzar.

En la part superior dreta, trobarem una icona d'un teclat, que al prémer sobre seu ens apareixerà un teclat virtual.

En la primera finestra trobem una sèrie d'instruccions que hem de realitzar abans de començar, com son: tenir ben connectades les pinces, i si es el cas de tenir connexió a Internet, configurar-la.

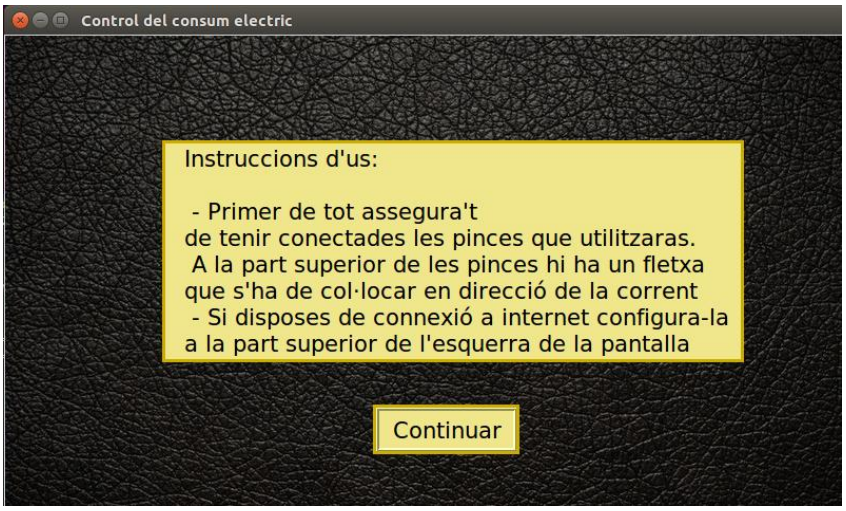

Figura 19: Finestra d'inici

Prement el boto de continuar accedim a la següent finestra on haurem d'elegir el tipus d'instal·lació ja sigui trifàsica o monofàsica.

L'única diferència és que si escollim trifàsic per defecte ja tindrem tres línies de mesura, en canvi en monofàsic podrem escollir entre una, dos o bé tres.

Figura 20: Finestra per escollir tipus d'instal·lació

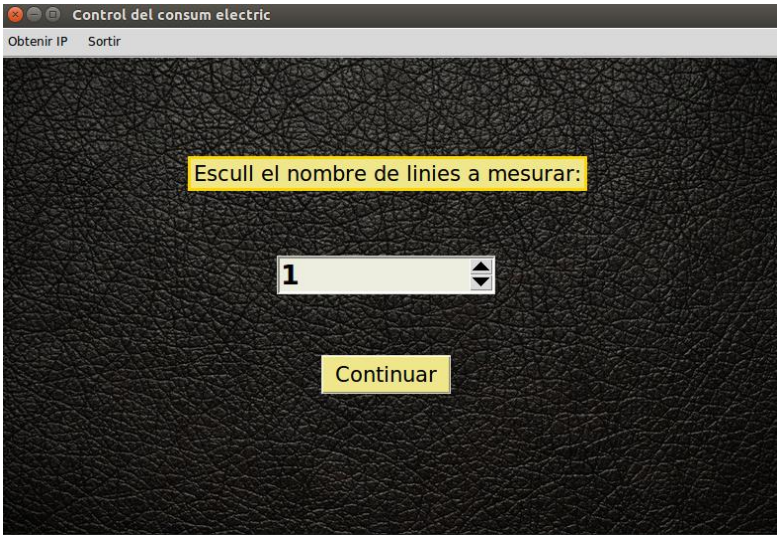

Figura 21: Finestra per escollir la quantitat de linies de mesura

En la següent finestra l'usuari podrà escollir la base de dades on vulgui guardar les lectures o bé crea'n una de nova.

Per defecte el temps entre lectures serà de **cinc segons** i el nom de cada linia de mesura Pinça 1, Pinça 2, i Pinça 3, tot i això es podrà canviar al gust que es vulgui.

Cal destacar que fins que no s'hagi elegit una base de dades, no apareixerà el boto per continuar.

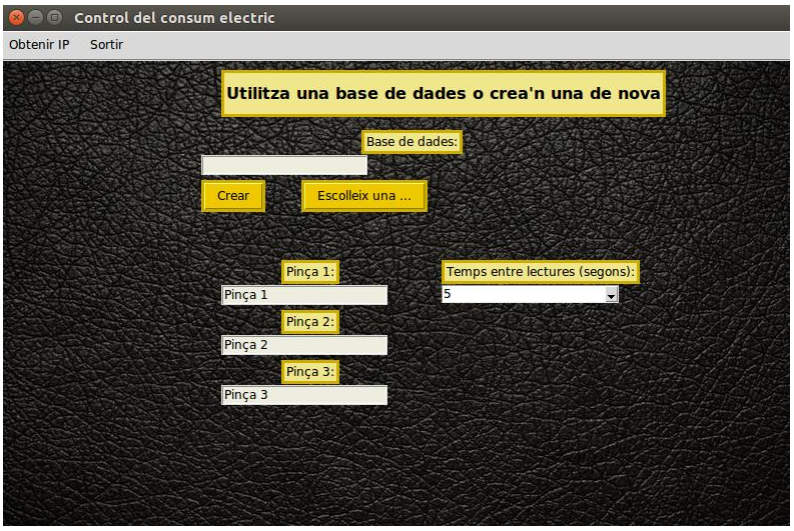

Figura 22: Finestra per la base de dades (1)

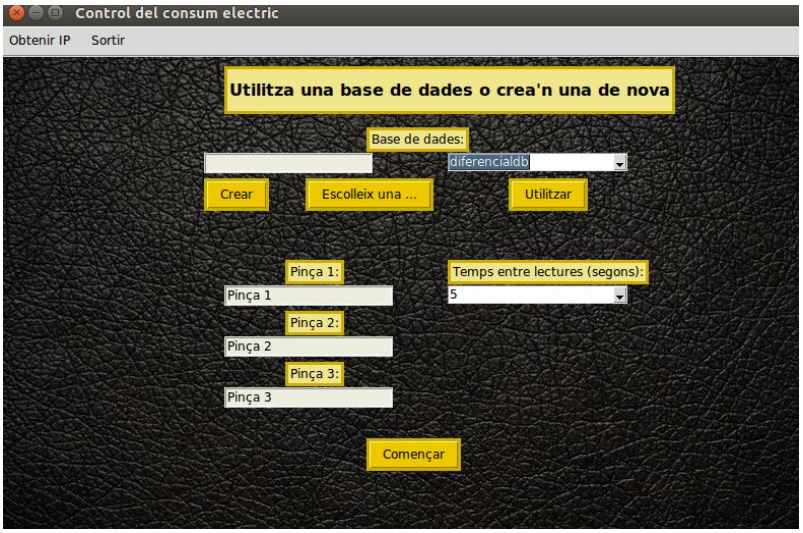

Figura 23: Finestra per la base de dades (2)

Finalment, en l'ultima finestra podem veure la potència instantània de cada línia de mesura.

En la part superior hi ha una barra menú en la qual trobem dos botons: un per sortir i l'altre per obtenir la IP. La IP la utilitzarem més endavant si volem veure les dades en la Plataforma Web

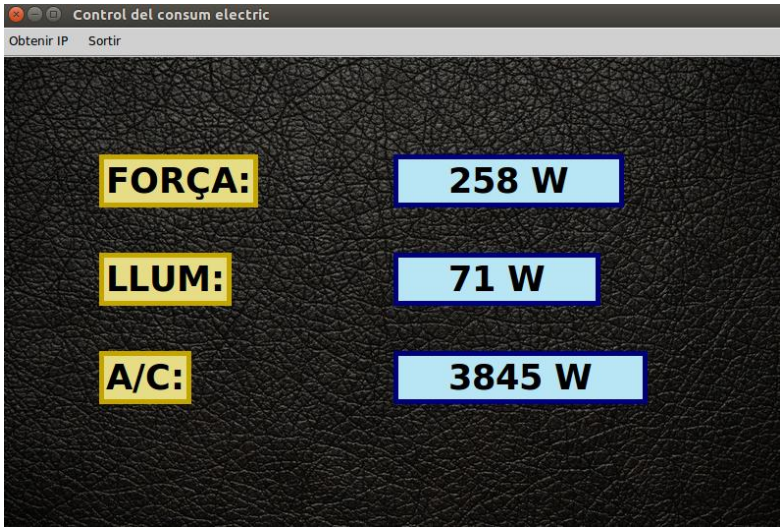

Figura 24: Finestra per a la visualització de dades

5.4 Plataforma web

Com hem dit anteriorment, s'ha optat per utilitzar el grafana ja que te una fàcil i senzilla interacció, a més a més, te una relació molt directa amb la base de dades InfluxDB.

Per al seu funcionament primer de tot ens hem d'assegurar de tenir connexió a internet (al pc i a la Raspberry) i posteriorment la seva instal·lació des de la seva pàgina web: *http://docs.grafana.org/installation/* , on podrem escollir el sistema operatiu. La descàrrega serà un arxiu .zip, que haurem de descomprimir i executar l'arxiu grafana-server.exe. A partir d'aquest punt ja podrem accedir a la plataforma web, sempre que haguem executat l'arxiu. Hi ha la possibilitat de configurar que l'arxiu grafana-server.exe s'executi cada cop que s'inicïi l'ordenador, això estalviaria haver d'executar l'arxiu manualment cada cop.

Un cop executat, podem accedir a la plataforma web obrint el navegador i escrivint la direcció ***http://localhost:3000*** on apareix una primera finestra on ens demana una identificació d'usuari i contrasenya que son admin i admin respectivament

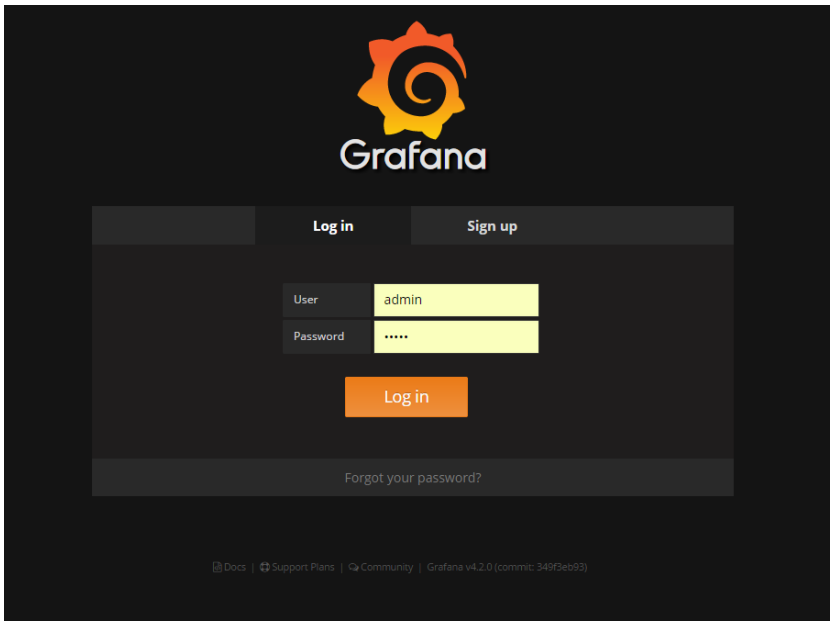

Figura 25: Pàgina d'inici Grafana

A partir d'aquest moment ja podem començar a configurar al nostre gust, i el primer que farem és establir quina base de dades utilitzarem. Per fer-ho anirem al menú de dalt a l'esquerra i entrarem en l'apartat de "Data Sources"

i pressionarem sobre de +Add data source i ens apareixerà la següent pantalla:

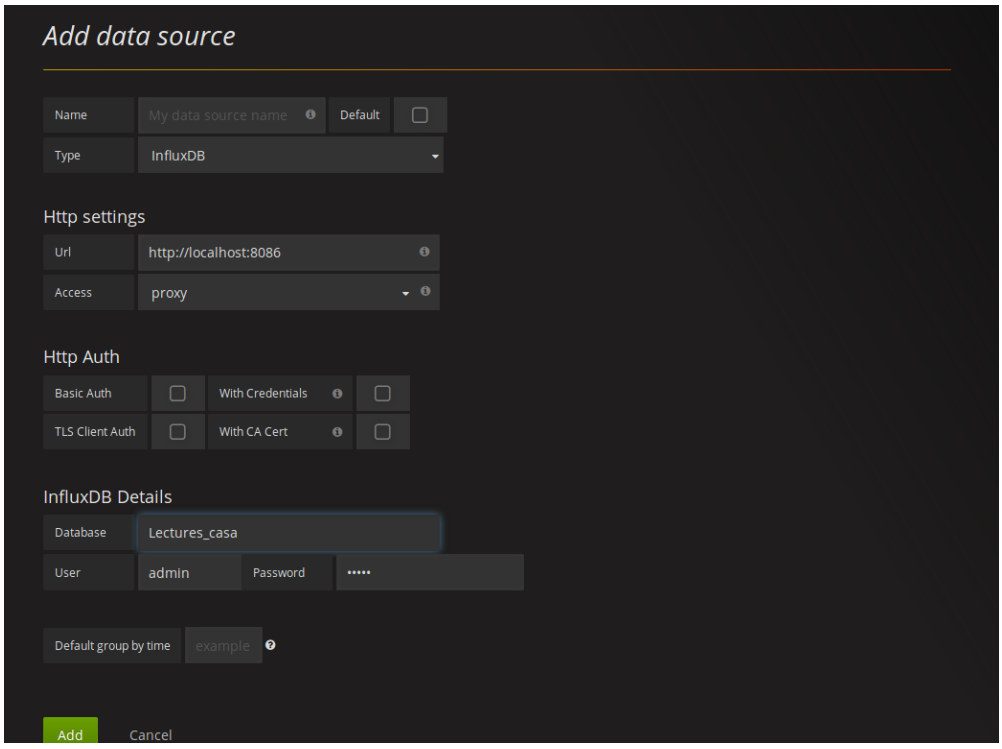

Figura 26: Configurar la base de dades

Aquí definirem diversos paràmetres:

Type: InfluxDB
Url: http://"IP Raspberry":8086
Acces: proxy
Database: Nom de la base de dades
User: admin
Password: admin

Seguidament premem el boto Add i ja tindrem la base de dades guardada al grafana. Cal recordar que per a que la base de dades estigui disponible hem de tenir la Raspberry connectada i amb connexió a internet.

Finalment l'últim pas es crear els gràfics al nostre gust en l'apartat del menú "Dashboards" i "New". Caldrà ajustar alguna configuració més, ho farem col·locant-nos sobre el gràfic on en apareixerà un requadre on pressionarem "edit".

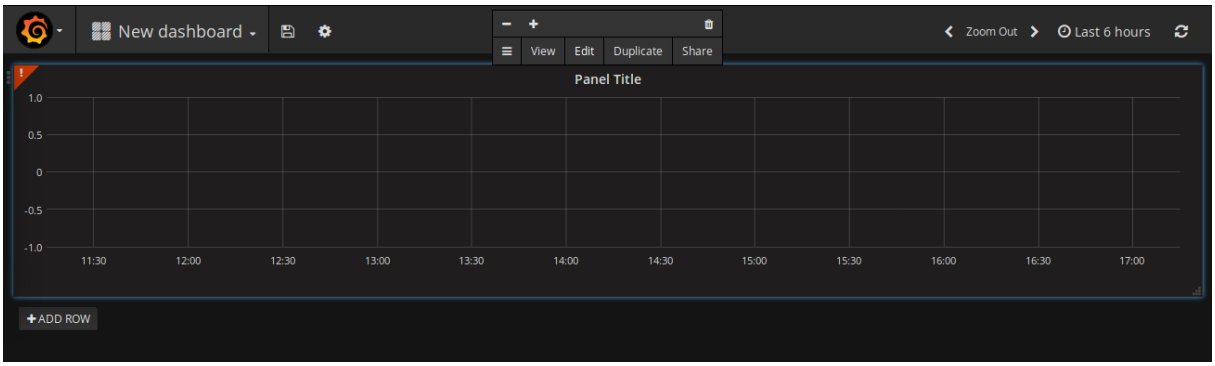

Figura 27: Creació d'un gràfic

En aquest apartat haurem de decidir de quina base de dades volem agafar les lectures, també la representació de les variables per la pestanya field (Pinça 1, Pinça 2, Pinça 3) i el temps de mostreig de dades és a dir, cada quan temps accedim a la base de dades per agafar els valors, que no es el temps de recopilació de dades del sensor.

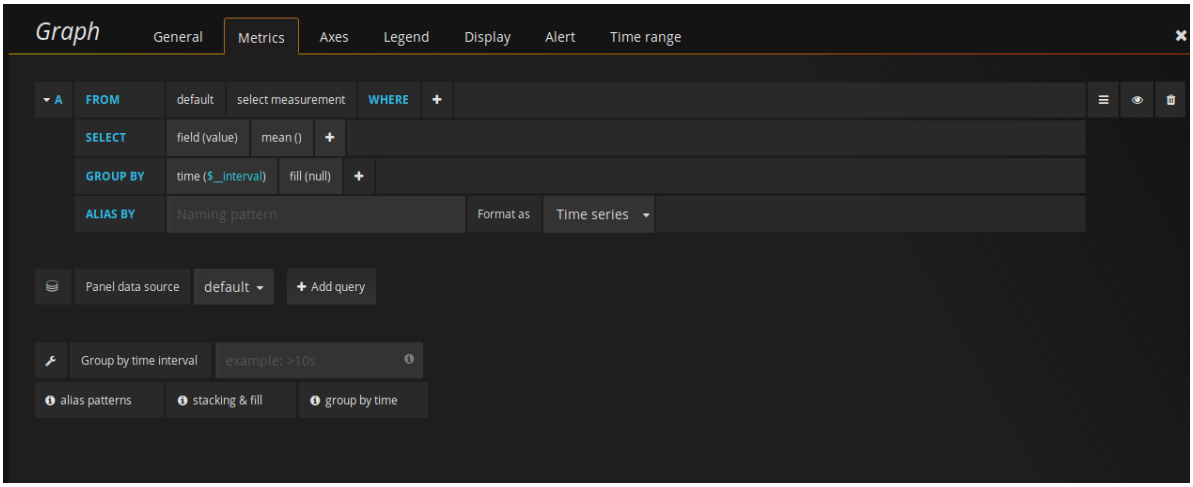

Figura 28: Configuració dels paràmetres del gràfic

A partir d'aquest punt ja sol queda crear tants gràfics, etiquetes, etc com es vulgui i edita'ls cadascú al seu gust.

6. Resultats

Després d'acabar el muntatge per complet, programar la Raspberry i instal·lar i crear els gràfics al Grafana, s'ha realitzat una prova del seu funcionament.

La prova ha estat connectar els sensors al quadre elèctric de casa en tres fils: un al interruptor de la il·luminació de la casa, el segon al interruptor diferencial que governa el A/C, la vitroceràmica i els congeladors, i finalment l'últim a la força, on es connecten tots els endolls

Els resultats els podem visualitzar mitjançant la pantalla incorporada, o bé per la plataforma web Grafana.

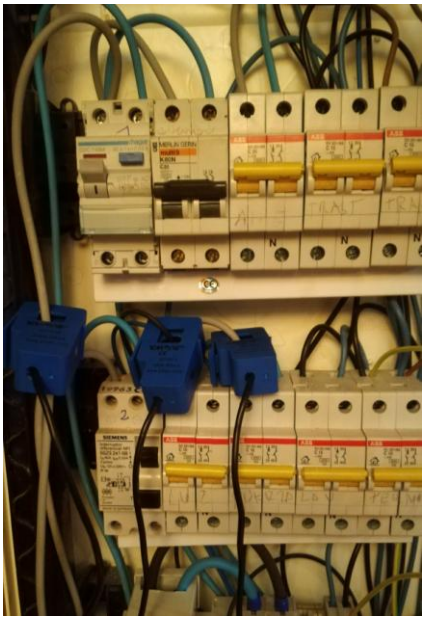

Figura 29: Col·locació de pinces al quadre elèctric

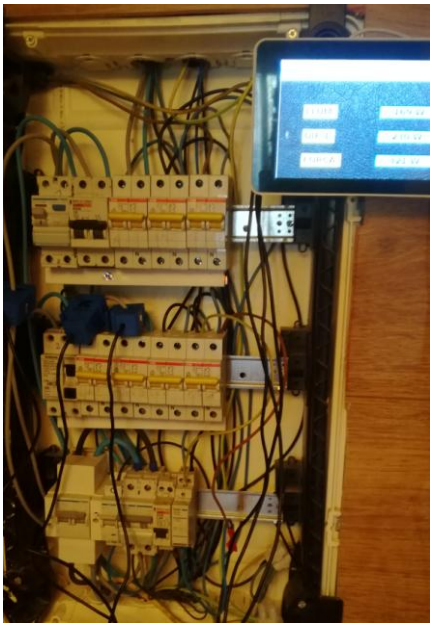

Figura 30: Muntatge al quadre elèctric

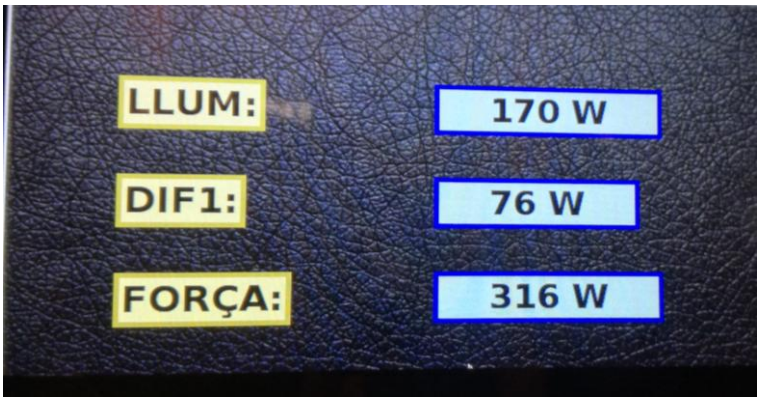

Figura 31: Visualització de les dades a la pantalla tàctil

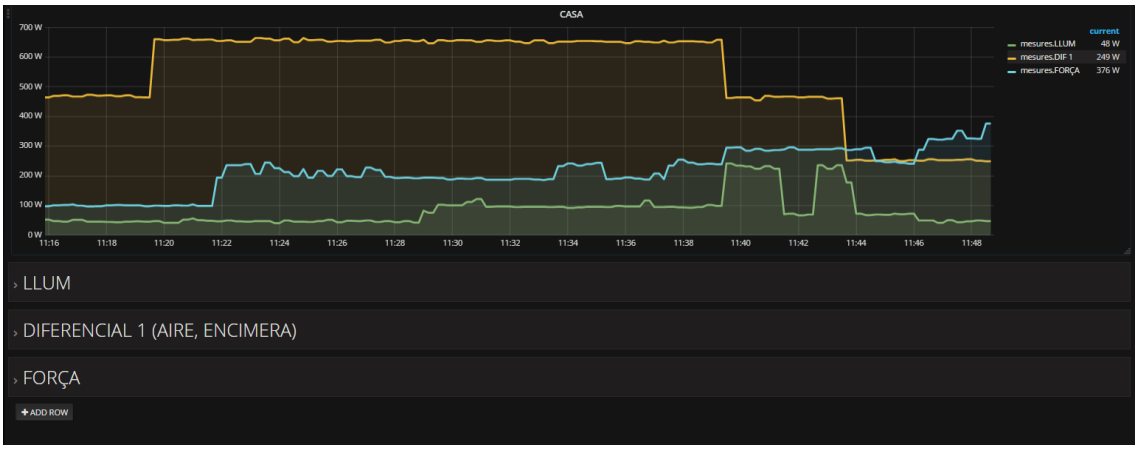

Figura 32: Visualització de les dades al Grafana (1)

Figura 33: Visualització de les dades al Grafana (2)

7. Pressupost

En aquest apartat s'ha elaborat el pressupost de tots els elements necessaris i utilitzats per a la realització del projecte.

Taula 2: Pressupost real del projecte

Material	Preu	Quantitat	Preu total
Raspberry Pi 3	41,95 €	1	41,95 €
Targeta SD 16GB	6,50 €	1	6,50 €
Font alimentació	8,00 €	2	16,00 €
RPICT3 - Raspberrypi SCT Adaptor	12,00 €	1	12,00 €
Transformador de correntSCT013 100A/50mA	7,00 €	3	21,00 €
Pantalla tàctil TouchScreen 7" + Carcassa	94,99 €	1	94,99 €
Ma d'obra	10,00 €	1	10,00 €
		TOTAL	**202,44 €**

En la següent taula s'ha fet un pressupost més detallat per poder llençar el producte al mercat en un futur adquirint els productes utilitzats a un preu més barat.

Taula 3: Pressupost possible del projecte

Material	Preu	Quantitat	Preu total
Raspberry Pi 3	37,00 €	1	37,00 €
Targeta SD 16GB	6,50 €	1	6,50 €
Font alimentació	3,90 €	2	7,80 €
RPICT3 - Raspberrypi SCT Adaptor	12,00 €	1	12,00 €
Transformador de correntSCT013 100A/50mA	4,65 €	3	13,95 €
Pantalla tàctil TouchScreen 7" + Carcassa	38,95 €	1	38,95 €
Ma d'obra	10,00 €	1	10,00 €
		TOTAL	**126,20 €**

Per tant en una inversió de 126,20€ podríem posar un preu de venta PVP:**200€.**

8. Conclusions

Les conclusions les dividirem en dos tipus: les tècniques, referent al projecte, i les personals, referents a l'aportació que ha tingut aquest projecte per mi.

8.1 Tècniques

Cal esmentar des d'un bon principi, que s'ha arribat a la idea final del que es buscava en un principi: monitoritzar el consum elèctric.

Referent al material utilitzat, cal dir que les sensacions son força bones, tot i la Raspberry ha estat una molt bona elecció per realitzar aquest projecte ja que té moltes funcionalitats, i es pot adaptar fàcilment a les necessitats desitjades tot i que no l'hem exprimit ni una mínima part del que pot arribar a fer. Per altra banda el RPICT3, ha estat un dels altres grans encerts, tot i que no podia ser perfecte, a l'hora de mesurar donar un error marginal d'uns 50W.

Basant-nos en que hem aconseguit el primer objectiu, el següent es veure si estem amb un producte, que tot i que encara que te moltes mancances en tema de programari, pot competir amb altres productes del mercat, ja sigui amb el preu, o bé perque ofereix altres prestacions. Per això hem realitzat una taula de comparació amb els altres productes.

Taula 4: Comparació amb altres prototips

CARACTERÍSTIQUES	Pack Efergy P2 + Engage Hub	Mirubox v2	EmonPi	Prototip
Consum elèctric	SI	SI	SI	SI
Consum energètic	SI	SI	SI	NO
Linies de mesura	3	1	2	3
Corrent màxima de mesura	100A	100A	100A	100 A
Monitor	LCD	NO	LCD	TOUCHSCREEN
Vida útil	12 mesos (piles)	-	-	
Alimentació	Piles o xarxa elèctrica	Xarxa elèctrica	Xarxa elèctrica	Xarxa elèctrica
Plataforma web	SI	SI	SI	SI
Ethernet	SI	NO	SI	SI
Wifi	NO	SI	SI	SI
Diferenciació de mesures	?	?	SI	SI
Aplicació movil	SI	SI	SI	NO
PVP	168,00 €	110,00 €	200,00 €	200, 00€

Analitzant el nostre prototip en comparació als altres, en el nostre cas no podem saber el consum elèctric, tot i que es podria crear un mètode en el firmware que ho fes, però no s'havia contemplat aquesta opció.

Els punts forts són que tenim connexió via Wifi i Ethernet en front a la majoria dels altres, i tot i no tenir connexió en el moment de les lectures, podrem obtenir-les i visualitzar-les a la plataforma en un altre moment que puguem connecta'ns a xarxa indistintivament de si estem o no mesurant les lectures,també que podem diferenciar les mesures realitzades, guardant-les en diferents variables i bases de dades. Per continuar amb el que ens avantatja els demès és la pantalla tàctil que farà molt minimalista l'ús del prototip.

8.2 Personals

Durant el desenvolupament del projecte han anat sortint diverses dificultats que han esdevingut pèrdues de temps bastant significants, la majoria basades en la inexperiència del món de la programació.

Tot i això estic molt satisfet d'haver aconseguit l'objectiu que em vaig marcar, el de realitzar una cosa exclusivament meva.

Aquest projecte m'ha ajudat a veure certes coses d'una forma diferent, una d'elles és la percepció de la figura de l'Enginyer, he entès que ser Enginyer electrònic, mecànic, ... no ens diferencia tant com creia al principi, al final el que adquirim és la capacitat de resoldre problemes i adversitats que van apareixent, l'altra és que és la única aplicació teòrica de tot el Grau d'Enginyeria Electrònica que he utilitzat en la pràctica, gràcies a les assignatures de Programació i Comunicacions he descobert el món de la Raspberry i la infinitat de projectes que és poden arribar a fer, també espero que en un futur la Raspberry és doni a conèixer més en àmbits més professionals.

9. Bibliografia

[1] http://guia-tkinter.readthedocs.io/es/develop/

[2] https://raspberryparatorpes.net/tag/primeros-pasos-raspberry-pi/

[3] https://guide.openenergymonitor.org/setup/

[4] http://www.mirubee.com/

[5] http://www.efimarket.com/pack-efergy-e2-trifasica-engage-hub

[6] http://lechacal.com/wiki/index.php/Main_Page

[7] http://influxdb-python.readthedocs.io/en/latest/

[8] http://docs.grafana.org/

ANNEX

lectures_sensor.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import serial

class PowerServer(object):

        def __init__(self):
            super(PowerServer, self).__init__()



        def read_sensor(self):

             while True:
                 try:

                    ser = serial.Serial('/dev/ttyAMA0', 38400, timeout=1)
                    #Obtenim resposta del port serial
                    response = ser.readline()
                    #separem resposta en comes
                    self.z = response.split(",")
                    #eliminem els salts de linia que puguin apareixer
                    self.z = [x.replace("\r\n","") for x in self.z]
                    #convertim els valors string en float
                    self.z = map(float, self.z)

                    #Convertim a int ja que influx només accepta int
                    for i in range(len(self.z)):
                         self.z[i]=int(self.z[i])
                    break
                except KeyboardInterrupt:
                    print "Lectura aturada per teclat"
                    ser.close()

if __name__ == "__main__":

        c = PowerServer()
        c.read_sensor()
```

database.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from influxdb import InfluxDBClient
from influxdb.client import InfluxDBClientError
import datetime
import time
from ast import literal_eval
from lectures_sensor import PowerServer

class Database(object):

        def __init__(self):
            super(Database, self).__init__()

            #Parametres oer defecte que definim a la base de dades
            self.powerserver = PowerServer()
            self.USER = 'root'
            self.PASSWORD = 'root'
            self.host ='localhost'
            self.port = 8086
            self.client = InfluxDBClient(self.host, self.port, self.USER,
                                         self.PASSWORD)

        def create_database(self, dbname):

            #Creem base de dades amb el nom que ens passa la funció
            self.DBNAME = dbname
            self.client.create_database(self.DBNAME)

        def insert_data(self):

                    #lectura del sensor
                    self.powerserver.read_sensor()

                    #Introduim les lectures a la base de dades
                    metric = "mesures"
                    series=[]
```

```
                now = datetime.datetime.today()
                pointValues = {
                        "measurement": metric,
                        "fields":{
                              "Pinça 1": self.powerserver.z[0],
                              "Pinça 2": self.powerserver.z[1],
                              "Pinça 3": self.powerserver.z[2]
                          },
                    }

                series.append(pointValues)

                self.client = InfluxDBClient(self.host, self.port,
                                             self.USER, self.PASSWORD,
                                             self.DBNAME)
                retention_policy = 'awesome_policy'
                self.client.create_retention_policy(retention_policy,
                                                    '20w', 3,
                                                     default=True)
                self.client.write_points(series,
                                         retention_policy=retention_policy)


        def get_database(self):

            #funció per obtenir les bases de dades guardades
            self.list_databases = []
            dbs = self.client.get_list_database()

            #el que ens retorna la funció ho posem en una llista i separat
            for x in dbs:
                for key, value in x.iteritems():
                    temp = [value]

                data = u", ".join(temp)
                self.list_databases.append(data)

if __name__ == "__main__":

        print " 'Control of electric consumption' "
        c = Database()
```

```
        c.create_database()
```

aplicacio_tkinter.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

from Tkinter import *
import Tkinter as tk
from tkFont import Font
import ttk
from lectures_sensor import PowerServer
from database import Database
import time
import commands
import os


class Tkinter(object):

        def __init__(self):
                super(Tkinter, self).__init__()

                #Inicialitzem la finestra emb les mides i el titol
                self.appl = Tk()
                self.appl.title("Control del consum electric")
                self.appl.geometry("800x450")
                self.fondo = PhotoImage(file="fondo.gif")
                self.lbl_Fondo = Label(self.appl,
                                        image=self.fondo).place(x=0,y=0)

                #Importem les classes dels altres arxius
                self.database = Database()
                self.sensor = PowerServer()

                #TIpus i mides de lletres que utilitzarem
                self.myFont = Font(family="Verdana", size=16)
                self.myFont2 = Font(family="Verdana",weight="bold", size=20)
                self.myFont3 = Font(family="Verdana",weight="bold", size=26)
                self.myFont4 = Font(family="Verdana", size=12, weight="bold")

                #Definim les variables de lectura i la seva estructura
```

```
self.sensor1 = StringVar()
self.sensor2 = StringVar()
self.sensor3 = StringVar()
self.sensor1.set("- - "+" W")
self.sensor2.set("- -"+" W")
self.sensor3.set("- -"+" W")

#Insertem el text de les instruccions a seguir
lbl = Label(self.app1,text=" Instruccions d'us:"
           "\n\n  - Primer de tot assegura't\n"
           " de tenir conectades les pinces que
             utilitzaras.\n"
           "  A la part superior de les pinces hi ha un
              fletxa\n"
           " que s'ha de col·locar en direcció de la
              corrent\n"
           "  - Si disposes de connexió a internet
               configura-la\n"
           " a la part superior de l'esquerra de la
               pantalla",
            justify=LEFT, font=self.myFont,bg="khaki",
            highlightbackground="gold3",
            highlightthickness=3, padx=10, )
lbl.place(x=150,y=100)

#Boto per continuar despres de llegir les instruccions
button_continuar = Button(self.app1,text="Continuar",
                          font=self.myFont,
                          command=self.tipus_instalacio,
                          bg="khaki", bd=3,
                          activebackground="gold",
                          relief=RIDGE,
                          overrelief=GROOVE,
                          highlightbackground="gold3",
                          highlightthickness=3 )
button_continuar.place(x=350,y=350)


self.app1.mainloop()
```

```
def tipus_instalacio(self):

        #Destruim finestra anterior i creem una nova
        self.hide_window("app1")
        self.new_window()

        #Nova etiqueta amb dos botons per elegir la instalació
        text = "Escull el tipus d'instalació"
        lbl = Label(self.app2,text=text, font=self.myFont, justify=
                    CENTER, bg="khaki",
                    padx=2,highlightbackground="gold",
                    highlightthickness=3)

        button_mono = Button(self.app2,text="MONOFÀSIC",
                             font=self.myFont2, bg="khaki",
                             bd=3, activebackground="gold",
                             relief=RIDGE,
                             overrelief=GROOVE, width=12,
                             height=3, command=self.monofasic)

        button_tri = Button(self.app2,text="TRIFÀSIC",
                            font=self.myFont2, bg="khaki",
                            bd=3, activebackground="gold",
                            relief=RIDGE,
                            overrelief=GROOVE, width=12,
                            height=3, command=self.trifasic)

        lbl.place(x=250, y=100)
        button_mono.place(x=150,y=270)
        button_tri.place(x=410,y=270)


def monofasic(self):

        #Destruim finestra anterior i creem una nova
        self.destroy_window("app2")
        self.new_window()
```

```
        def linies():

                #passem el valor obtingut a un enter i cridem la
                 seguent funció
                nlinies = self.spin_linies.get()
                self.linies = int(nlinies)
                self.eleccio_base_dades()


        #Etiqueta i Spin per escollir el nombre de linies
          monofàsiques a mesurar
        text = "Escull el nombre de linies a mesurar:"
        lbl = Label(self.app2,text=text, font=self.myFont,
                    justify= CENTER, bg="khaki", padx=2,
                    highlightbackground="gold", highlightthickness=3)

        self.spin_linies = Spinbox(self.app2, font=self.myFont2,
                                   bg="ivory2", from_=1, to=3,
                                   width=10, borderwidth=2)

        button_continuar = Button(self.app2, text="Continuar",
                                  font=self.myFont,
                                  bg="khaki",activebackground="gold",
                                  command=linies)

        lbl.place(x=190,y=100)
        self.spin_linies.place(x=280,y=200)
        button_continuar.place(x=325, y=300)


def trifasic(self):

        #Defenim que a trifasic hi haurà 3 linies obligatories
        self.linies = 3
        self.eleccio_base_dades()


def eleccio_base_dades(self):

        #Destruim finestra anterior i creem una nova
```

```
self.destroy_window("app2")
self.new_window()

def nova_database():

        #Obtenim el valor entrat de la nova base de dades
        self.base = self.txtbase.get()
        #Creem la nova base de dades cridant la funcio de
         l'arxiu database.py
        self.database.create_database(self.base)
        #Cridar funció per seleccioar la base dades a
          utilitzar
        llista_base_dades()


def llista_base_dades():

        #Actualitzem totes les bases de dades
        self.database.get_database()

        #Desplegable amb totes les bases de dades i boto per
          seleccionar
        self.desplegable = ttk.Combobox(self.app2)
        self.desplegable['values'] =
                              self.database.list_databases
        self.desplegable.current(0)

        button_utilitzar = Button(self.app2,text="Utilitzar",
                                  bg="gold2",
                                  command=vella_database,
                                  highlightbackground="gold3",
                                  highlightthickness=3,
                                  activebackground="gold")

        self.desplegable.place(x=440,y=95)
        button_utilitzar.place(x=500,y=120)

def vella_database():

        #Obtenim el valor selecconat al desplegable i cridem
          a la seguent funció
```

```
        self.base = self.desplegable.get()
        utilitzar_database()


def utilitzar_database():

        #Creem la base de dades per poderla utilitzar, amb el
         valor obtingut anterior
        self.database.create_database(self.base)

        #Boto per començar les lectures
        button_comen = Button(self.app2,text="Començar",
                              command=self.lectura_dades,
                              highlightbackground="gold3",
                              highlightthickness=3,
                              activebackground="gold",
                              bg="gold2")

        button_comen.place(x=360,y=375)



#Etiquetes, botons i entrades de text per escollir la base de
  dades
text = "Utilitza una base de dades o crea'n una de nova"
lbl = Label(self.app2,text=text, bg="khaki", justify= CENTER,
            pady=10, font=self.myFont4,
            highlightbackground="gold3",
            highlightthickness=3)

#Variable a la qual assignarem el nom de la base de dades
self.entradabase = StringVar()

lbl_bd = Label(self.app2,text="Base de dades:", bg="khaki",
               highlightbackground="gold3",
               highlightthickness=3)

#Entrada de text per la base de dades
self.txtbase = Entry(self.app2,
                     textvariable=self.entradabase,
                     bg="ivory2")
```

```
#Boto per crear la base de dades
button_crear = Button(self.app2,text="Crear",
                      bg="gold2",command=nova_database,
                      highlightbackground="gold3",
                      highlightthickness=3,
                      activebackground="gold")

#Boto per escollir una base de dades ja guardada
button_escollir = Button(self.app2,text="Escolleix una ...",
                         bg="gold2",
                         command=llista_base_dades,
                         highlightbackground="gold3",
                         highlightthickness=3,
                         activebackground="gold")

lbl_temps_lectura = Label(self.app2,text="Temps entre
                          lectures (segons):", bg="khaki",
                          highlightbackground="gold3",
                          highlightthickness=3)
self.temps_lectura = ttk.Combobox(self.app2)
self.temps_lectura['values']=['5','10','15','30',
                              '60','300','600','900']
self.temps_lectura.current(0)



lbl.place(x=220, y=10)
lbl_bd.place(x=360,y=70)
self.txtbase.place(x=200, y=95)
button_crear.place(x=200,y=120)
button_escollir.place(x=300,y=120)
lbl_temps_lectura.place(x=440,y=200)
self.temps_lectura.place(x=440,y=225)



#Etiquetes i entrades de text per el nom de cada linia de
 mesura
#segons les linies que s'hagin elegit anteriorment
```

```
if self.linies <= 3:

    lbl_pin1 = Label(self.app2,text="Pinça 1:",bg="khaki",
                     highlightbackground="gold3",
                     highlightthickness=3)

    self.entrada1 = StringVar(self.app2, 'Pinça 1')

    self.txtpin1 = Entry(self.app2,
                         textvariable=self.entrada1,
                         bg="ivory2")

    lbl_pin1.place(x=280,y=200)
    self.txtpin1.place(x=220,y=225)


if self.linies != 1:

    lbl_pin2 = Label(self.app2,text="Pinça 2:",bg="khaki",
                     highlightbackground="gold3",
                     highlightthickness=3)

    self.entrada2 = StringVar(self.app2, 'Pinça 2')

    self.txtpin2 = Entry(self.app2,
                         textvariable=self.entrada2,
                         bg="ivory2")

    lbl_pin2.place(x=280,y=250)
    self.txtpin2.place(x=220,y=275)

if self.linies == 3:

    lbl_pin3 = Label(self.app2,text="Pinça 3:",bg="khaki",
                          highlightbackground="gold3",
                          highlightthickness=3)

    self.entrada3 = StringVar(self.app2, 'Pinça 3')
```

```
            self.txtpin3 = Entry(self.app2,
                                 textvariable=self.entrada3,
                                 bg="ivory2")

            lbl_pin3.place(x=280,y=300)
            self.txtpin3.place(x=220,y=325)


def lectura_dades(self):

    #Guardem la variable del temps entre lectures en una variable
     entera
    self.temps = int(self.temps_lectura.get())

    self.destroy_window("app2")
    self.new_window()

    #Creem les etiquetes on es mostraran les dades en funcio de les
     linies elegides
    if self.linies <= 3:

            linia1 = Label(self.app2,text=self.entrada1.get()+':',
                           font=self.myFont3,
                           highlightbackground="gold3",
                           highlightthickness=5, bg="khaki")

            self.lectura1 = Label(self.app2,text=self.sensor1.get(),
                                  font=self.myFont3,
                                  padx=50, highlightbackground="navy",
                                  highlightthickness=5,
                                  bg="LightBlue1")

            linia1.place(x=100,y=100)
            self.lectura1.place(x=400,y=100)

    if self.linies != 1:

            linia2 = Label(self.app2,text=self.entrada2.get()+':',
                           font=self.myFont3,
                           highlightbackground="gold3",
```

```
                                highlightthickness=5, bg="khaki")

            self.lectura2 = Label(self.app2,text=self.sensor2.get(),
                                  font=self.myFont3,
                                  padx=50,
                                  highlightbackground="navy",
                                  highlightthickness=5,
                                  bg="LightBlue1")

            linia2.place(x=100, y=200)
            self.lectura2.place(x=400, y=200)

        if self.linies == 3:

            linia1 = Label(self.app2,text=self.entrada3.get()+':',
                                 font=self.myFont3,
                                 highlightbackground="gold3",
                                 highlightthickness=5, bg="khaki")

            self.lectura3 = Label(self.app2,text=self.sensor3.get(),
                                  font=self.myFont3,
                                  padx=50,
                                  highlightbackground="navy",
                                  highlightthickness=5,
                                  bg="LightBlue1")

            linia1.place(x=100, y=300)
            self.lectura3.place(x=400, y=300)

        #Cridem la funció que ens mostrarà les lectures
        self.actualitzar_dades()

    def actualitzar_dades(self):

            #Cridem la funció per insertar les lectures a la base de
             dades
            self.database.insert_data()
            #Cridem la funció per obtenir les lectures del sensor
            self.sensor.read_sensor()

            #Actualitsem les dades amb les lectures actuals
```

```
            if self.linies <= 3:
                self.sensor1.set(str(self.sensor.z[0])+" W")
                self.lectura1.config(text=self.sensor1.get())
            if self.linies != 1:
                self.sensor2.set(str(self.sensor.z[1])+" W")
                self.lectura2.config(text=self.sensor2.get())
            if self.linies == 3:
                self.sensor3.set(str(self.sensor.z[2])+" W")
                self.lectura3.config(text=self.sensor3.get())

            #Eecutem aquesta funció cada x segons
            self.app2.after(self.temps*1000, self.actualitzar_dades)




#////////////////////////////////////////////////////////////////////////

    def new_window(self):

        #funció per crear la finestra
        def ip():


            self.app_ip = Toplevel()
            self.app_ip.title("Adreça IP")
            self.app_ip.geometry("200x150")

            ip_adress = commands.getoutput ("ifconfig lo | grep inet |
                                            awk '{ print $2 }'")

            lbl = Label(self.app_ip,text=ip_adress, font=self.myFont,
                        justify= CENTER, bg="khaki",
                        padx=2,highlightbackground="gold",
                        highlightthickness=3)
            lbl.pack()

            boton = ttk.Button(self.app_ip, text='Sortir',
                               command=self.app_ip.destroy)
            boton.pack(side=BOTTOM, padx=20, pady=20)
```

```
    def exit():
        self.destroy_window(self.app1)

    self.app2 =tk.Toplevel()
    self.app2.title("Control del consum electric")
    self.app2.geometry("800x480")
    self.fondo2 = PhotoImage(file="fondo.gif")
    self.lbl_Fondo2 = Label(self.app2,
                            image=self.fondo2).place(x=0,y=0)

    self.barraMenu = Menu(self.app2)

    self.app2.config(menu=self.barraMenu)


    self.barraMenu.add_command(label="Obtenir IP", command=ip)
    self.barraMenu.add_command(label="  Sortir", command=exit)


    #self.app2.mainloop()

def destroy_window(self, finestra):

    #funció per destruir la finestra
    if finestra == "app1":
        self.app1.destroy()
    if finestra == "app2":
        self.app2.destroy()


def hide_window(self, finestra):

    #funció per amagar la fiestra
    if finestra == "app1":
        self.app1.withdraw()
    if finestra == "app2":
        self.app2.withdraw()
```

```
        def mostrar_finestra(self, finestra):

            #funció per mostrar la finestra
            if finestra == "app1":
                self.app1.deiconify()
            if finestra == "app2":
                self.app2.deiconify()



if __name__ == "__main__":

        print " 'Control of electric consumption' "
        c = Tkinter()
        c.monofasic()
```